

acrack.exe | crackmes.one

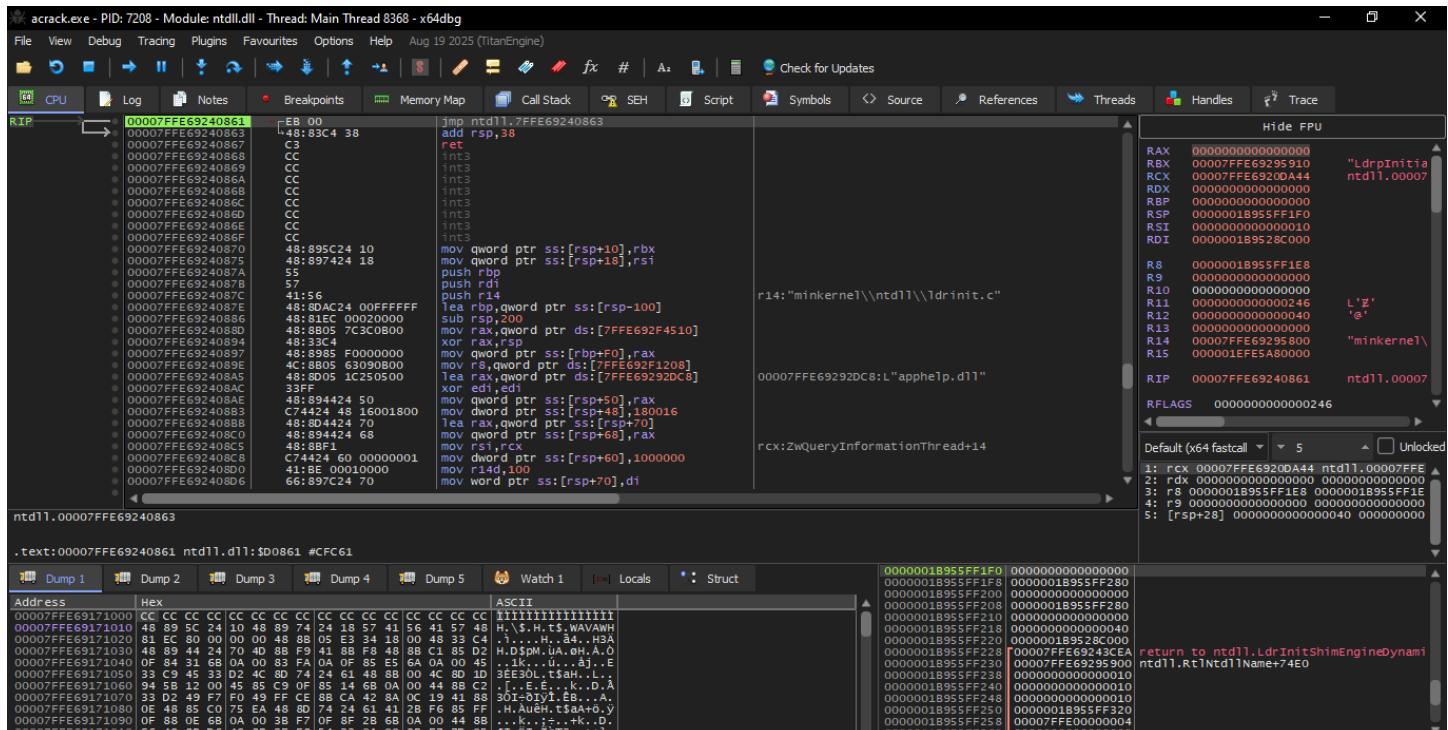
EASY LEVEL REVERSE ENGINEERING

So, this a very easy level RE practice for Reverse engineers Based in C. First of all, we want to see what's the application do. So, I just executed it through cmd.

```
C:\Windows\System32\cmd.exe
C:\Users\Piyusha Akash\Desktop\RevEng>acrack.exe
Mini crackme by @microsoftstorage writed on C!
Nickname: something
bad nickname! try again

C:\Users\Piyusha Akash\Desktop\RevEng>
```

After executing the application, we can see it will ask for Nickname and I just entered "something" as the Nickname and it will give me "bad nickname! Try again." So, we can see it is the wrong Nickname. Let's start Reverse engineer this application. So, in that case I use x64dbg as my tool for reverse this application. I just drag and drop the application to x64dbg.



So, Now I need to looking for strings in this application. So, I will right click the assembly area and go as this. Right click → Search for → All modules → String references. So, I can now see the all strings in this application.

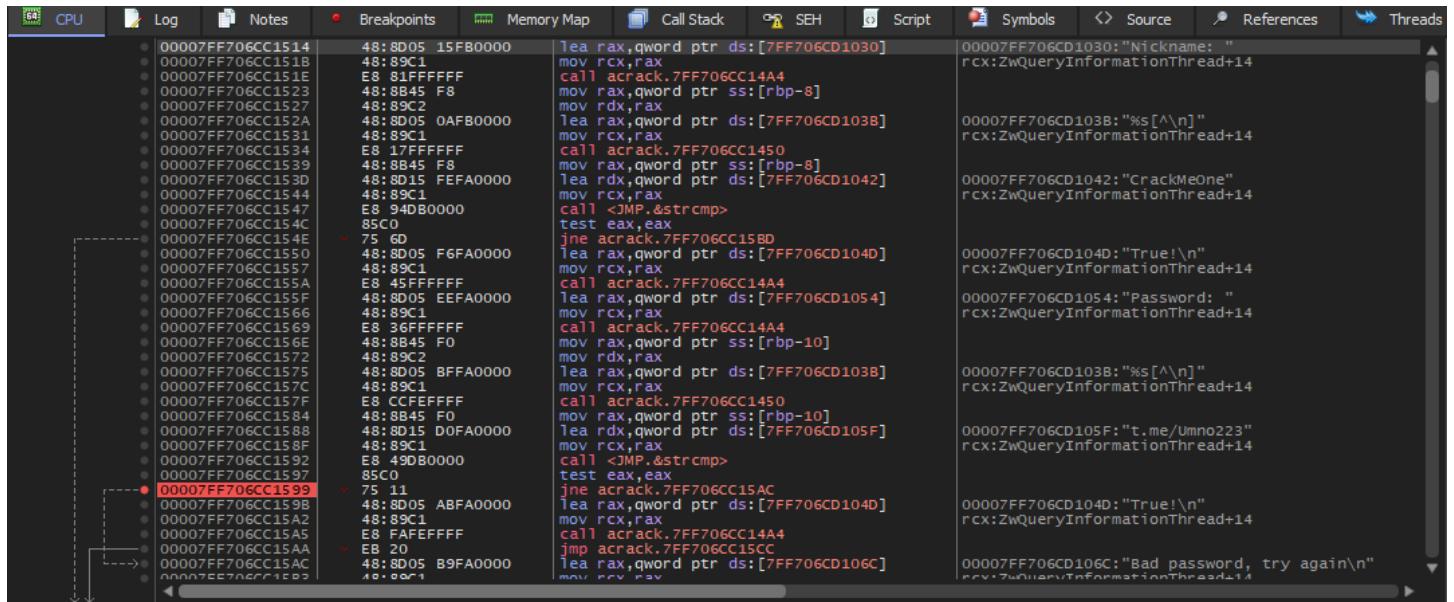
Address	Disassembly	String Address	String
00007FF706CC1505	lea rax,qword ptr ds:[7FF706CD1000]	00007FF706CD1000	"Mini crackme by @microsoftstorage writed on C!\n"
00007FF706CC1514	lea rax,qword ptr ds:[7FF706CD1030]	00007FF706CD1030	"Nickname: "
00007FF706CC152A	lea rax,qword ptr ds:[7FF706CD1038]	00007FF706CD1038	"\$s(^n)"
00007FF706CC153A	lea rax,qword ptr ds:[7FF706CD1045]	00007FF706CD1045	"CrackMeOne"
00007FF706CC1550	lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"True!\n"
00007FF706CC155F	lea rax,qword ptr ds:[7FF706CD1054]	00007FF706CD1054	"Password: "
00007FF706CC157F	lea rdx,qword ptr ds:[7FF706CD1038]	00007FF706CD1038	"\$s(^n)"
00007FF706CC158B	lea rdx,qword ptr ds:[7FF706CD105F]	00007FF706CD105F	"t.me/Umno223"
00007FF706CC1590	lea rax,qword ptr ds:[7FF706CD1060]	00007FF706CD1060	"Bad password, try again\n"
00007FF706CC15AC	lea rax,qword ptr ds:[7FF706CD106C]	00007FF706CD106C	"Bad nickname! try again\n"
00007FF706CC15BD	lea rax,qword ptr ds:[7FF706CD1085]	00007FF706CD1085	"Argument singularity (SIGN)"
00007FF706CC17C8	lea rbx,qword ptr ds:[7FF706CD111F]	00007FF706CD111F	"._matherr(): %s (%g, %g), (retval=%g)\n"
00007FF706CC17F7	lea rdx,qword ptr ds:[7FF706CD1118]	00007FF706CD1118	"Argument domain error (DOMAIN)"
00007FF706CC1830	lea rbx,qword ptr ds:[7FF706CD1140]	00007FF706CD1140	"Total loss of precision (LOSS)"
00007FF706CC1850	lea rbx,qword ptr ds:[7FF706CD1140]	00007FF706CD1140	"Overflow and error (OVERFLOW)"
00007FF706CC1870	lea rbx,qword ptr ds:[7FF706CD1188]	00007FF706CD1188	"The result is too small to be represented (UNDERFLOW)"
00007FF706CC187E	lea rdx,qword ptr ds:[7FF706CD11E6]	00007FF706CD11E6	"Total loss of significance (TLOSS)"
00007FF706CC18D0	lea rcx,qword ptr ds:[7FF706CD1240]	00007FF706CD1240	"Unknown error"
00007FF706CC18E0	lea rcx,qword ptr ds:[7FF706CD1240]	00007FF706CD1240	"MinGW-w64 APIProtect failed with code 0x00"
00007FF706CC1A51	lea rcx,qword ptr ds:[7FF706CD1280]	00007FF706CD1280	"VirtualQuery failed for %d bytes at address %p"
00007FF706CC1A60	lea rcx,qword ptr ds:[7FF706CD1260]	00007FF706CD1260	"Address %p has no image-section"
00007FF706CC1C10	lea rcx,qword ptr ds:[7FF706CD1348]	00007FF706CD1348	"%d bit pseudo relocation at %p out of range, targeting %p, yielding the value %p.\n"
00007FF706CC1DC4	lea rcx,qword ptr ds:[7FF706CD1318]	00007FF706CD1318	"% Unknown pseudo relocation bit size %d.\n"
00007FF706CC1E00	lea rdx,qword ptr ds:[7FF706CD1318]	00007FF706CD1318	"% Unknown pseudo relocation protocol version %d.\n"
00007FF706CC1E83	lea rsi,qword ptr ds:[7FF706CD13D1]	00007FF706CD13D1	"%?!"
00007FF706CC5512	lea rax,qword ptr ds:[7FF706CD13DA]	00007FF706CD13DA	"%?!"
00007FF706CC5547	lea rcx,qword ptr ds:[7FF706CD13DA]	00007FF706CD13DA	"%?!"
00007FF706CC58C8	lea rax,qword ptr ds:[7FF706CD13D6]	00007FF706CD13D6	"nan"
00007FF706CC5903	lea rax,qword ptr ds:[7FF706CD13D6]	00007FF706CD13D6	"nan"
00007FF706CC5940	lea rax,qword ptr ds:[7FF706CD13D6]	00007FF706CD13D6	"nan"
00007FF706CC5950	lea rax,qword ptr ds:[7FF706CD1690]	00007FF706CD1690	"%?!"
00007FF706CC5956	lea rdx,qword ptr ds:[7FF706CD1690]	00007FF706CD1690	"%?!"
00007FF706CC5991	lea rdx,qword ptr ds:[7FF706CD16AA]	00007FF706CD16AA	"%?!"
00007FF706CC8B4E	lea rax,qword ptr ds:[7FF706CD1698]	00007FF706CD1698	L"(null)"
00007FF706CC8B88	lea rdx,qword ptr ds:[7FF706CD16A6]	00007FF706CD16A6	"Name"
00007FF706CC8B96	lea rdx,qword ptr ds:[7FF706CD16A6]	00007FF706CD16A6	"%"
00007FF706CC8B9A	lea rdx,qword ptr ds:[7FF706CD16A6]	00007FF706CD16A6	"%"
00007FF706CC8B9D	lea rdx,qword ptr ds:[7FF706CD16A6]	00007FF706CD16A6	"%"
00007FF706CC8C978	lea rcx,qword ptr ds:[7FF706CD1820]	00007FF706CD1820	"Infinity"
00007FF706CCB0C7	lea rdx,qword ptr ds:[7FF706CD1A29]	00007FF706CD1A29	"an"
00007FF706CCD2B0	lea rdx,qword ptr ds:[7FF706CD1A20]	00007FF706CD1A20	"nf"
00007FF706CCD2A4	lea rdx,qword ptr ds:[7FF706CD1A23]	00007FF706CD1A23	"inty"
00007FF706CCD2A8	lea rdx,qword ptr ds:[7FF706CD1A23]	00007FF706CD1A23	"0x4545454545454545"
00007FF706CCD273	lea rdx,qword ptr ds:[7FF706CD1B8B]	00007FF706CD1B8B	"abcdef"
00007FF706CCD2A3	lea rdx,qword ptr ds:[7FF706CD1B82]	00007FF706CD1B82	"ABCDEF"
00007FF706CD12AD	and byte ptr gs:[7FF706CD1323],ah	00007FF706CD1323	"seudo relocation bit size %d.\n"
00007FF706CD157	add byte ptr gs:[7FF706CD2B2C],al	00007FF706CD2B2C	"_R4" ...

In this section I can see all the string and I will be looking for some hint. And I saw something. I found the password though strings easily. Let's try these credentials.

Address	Disassembly	String Address	String
706CC1505	lea rax,qword ptr ds:[7FF706CD1000]	00007FF706CD1000	"Mini crackme by @microsoftstorage writed on C!\n"
706CC1514	lea rax,qword ptr ds:[7FF706CD1030]	00007FF706CD1030	"Nickname: "
706CC152A	lea rax,qword ptr ds:[7FF706CD1038]	00007FF706CD1038	"\$s(^n)"
706CC153D	lea rdx,qword ptr ds:[7FF706CD1042]	00007FF706CD1042	"CrackMeOne"
706CC1550	lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"True!\n"
706CC155F	lea rax,qword ptr ds:[7FF706CD1054]	00007FF706CD1054	"Password: "
706CC1575	lea rax,qword ptr ds:[7FF706CD1054]	00007FF706CD1054	"\$s(^n)"
706CC1588	lea rdx,qword ptr ds:[7FF706CD105F]	00007FF706CD105F	"t.me/Umno223"
706CC159B	lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"True!\n"
706CC15BD	lea rax,qword ptr ds:[7FF706CD1061]	00007FF706CD1061	"Bad password. try again\n"
706CC17C8	lea rdx,qword ptr ds:[7FF706CD111F]	00007FF706CD111F	C:\Windows\System32\cmd.exe
706CC1840	lea rdx,qword ptr ds:[7FF706CD1140]	00007FF706CD1140	Mini crackme by @microsoftstorage writed on C!
706CC1860	lea rdx,qword ptr ds:[7FF706CD1140]	00007FF706CD1140	"Nickname: CrackMeOne
706CC187C	lea rdx,qword ptr ds:[7FF706CD1140]	00007FF706CD1140	True!
706CC1880	lea rdx,qword ptr ds:[7FF706CD1140]	00007FF706CD1140	Password: t.me/Umno223
706CC1A51	lea rdx,qword ptr ds:[7FF706CD105F]	00007FF706CD105F	True!
706CC1CD4	lea rcx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC1DD0	lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"0x4545454545454545"
706CC4B33	lea rsi,word	00007FF706CD104D	"0x4545454545454545"
706CC5547	lea rcx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC58C8	lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC5903	lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC634D	lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC6B56	lea rax,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC8919	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC8B86	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC8D96	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CC9378	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCBD7	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCD2B0	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCD2A4	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCD2A8	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCD2A9	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCD2A9	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCD2A9	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCD2A9	lea rdx,qword ptr ds:[7FF706CD104D]	00007FF706CD104D	"\$s(^n)"
706CCD12AD	and byte ptr gs:[7FF706CD1323],ah	00007FF706CD1323	"seudo relocation bit size %d.\n"
706CDB157	add byte ptr gs:[7FF706CD2B2C],al	00007FF706CD2B2C	"_R4" ...

So, we can see Nickname is CrackMeOne and Password is t.me/Umno223. These credentials are correct and we will get True! Message.

We found the password and nickname but it's not the game. What happen if we can crack the program for return 0 (True! message) for any string. Let's start. So, I just double click the "Nickname" string and then I will go to disassembled section matching to String.

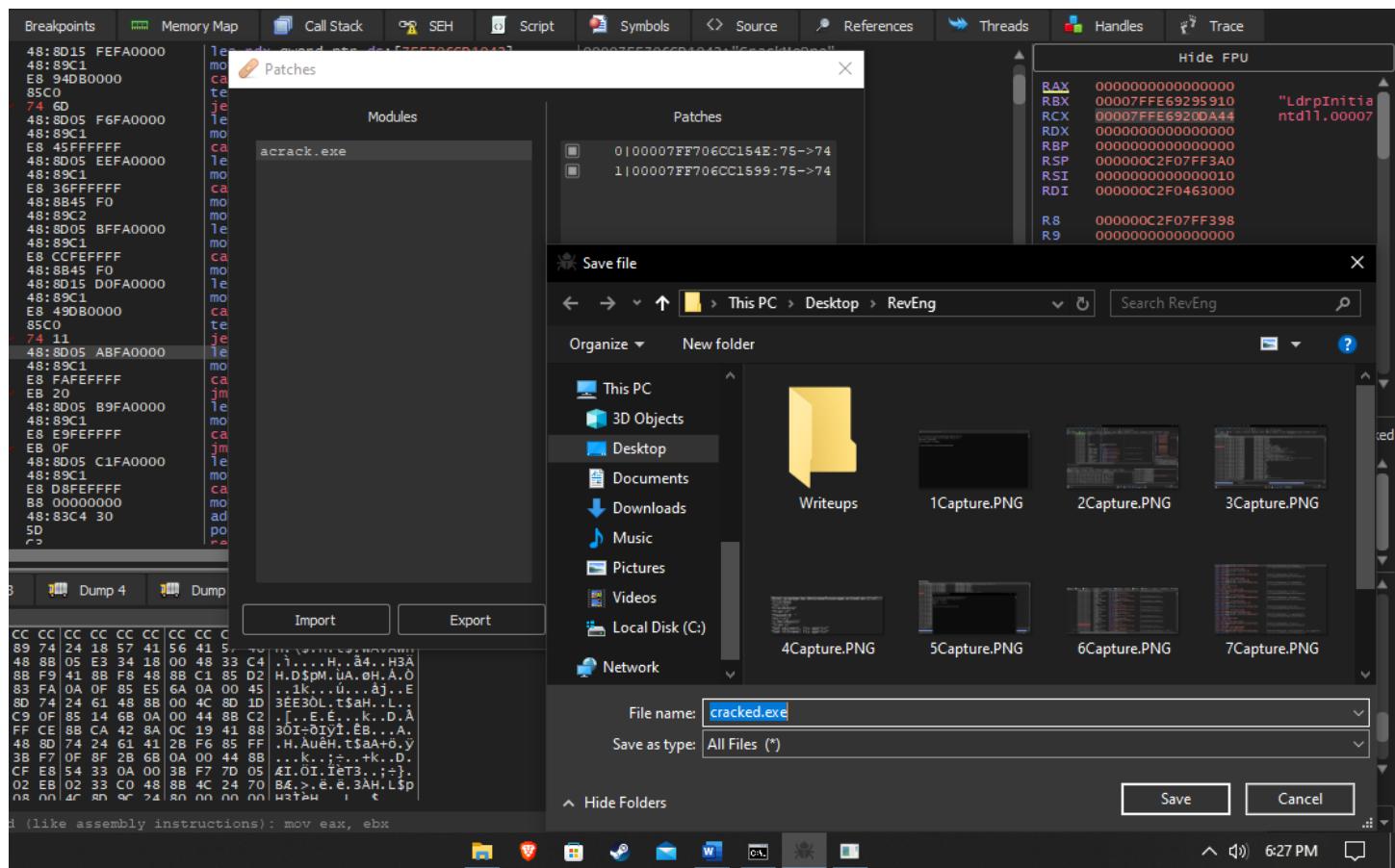


<pre> lea rax,qword ptr ds:[7FF706CD1030] mov rcx,rax call acrack.7FF706CC14A4 mov rax,qword ptr ss:[rbp-8] mov rdx,rax lea rax,qword ptr ds:[7FF706CD103B] mov rcx,rax call acrack.7FF706CC1450 mov rax,qword ptr ss:[rbp-8] lea rdx,qword ptr ds:[7FF706CD1042] mov rcx,rax call <JMP.&strcmp> test eax,eax jne acrack.7FF706CC15BD lea rax,qword ptr ds:[7FF706CD104D] mov rcx,rax call acrack.7FF706CC14A4 lea rax,qword ptr ds:[7FF706CD1054] mov rcx,rax call acrack.7FF706CC14A4 mov rax,qword ptr ss:[rbp-10] mov rdx,rax lea rax,qword ptr ds:[7FF706CD103B] mov rcx,rax call acrack.7FF706CC1450 mov rax,qword ptr ss:[rbp-10] lea rdx,qword ptr ds:[7FF706CD105F] mov rcx,rax call <JMP.&strcmp> test eax,eax jne acrack.7FF706CC15AC lea rax,qword ptr ds:[7FF706CD104D] mov rcx,rax call acrack.7FF706CC14A4 jmp acrack.7FF706CC15CC lea rax,qword ptr ds:[7FF706CD106C] mov rcx,rax </pre>	00007FF706CD1030:"Nickname: " rcx:ZwQueryInformationThread+14 00007FF706CD103B:"%s[\n]" rcx:ZwQueryInformationThread+14 00007FF706CD1042:"CrackMeOne" rcx:ZwQueryInformationThread+14 00007FF706CD104D:"True!\n" rcx:ZwQueryInformationThread+14 00007FF706CD1054:"Password: " rcx:ZwQueryInformationThread+14 00007FF706CD103B:"%s[\n]" rcx:ZwQueryInformationThread+14 00007FF706CD105F:"t.me/Umn0223" rcx:ZwQueryInformationThread+14 00007FF706CD104D:"True!\n" rcx:ZwQueryInformationThread+14 00007FF706CD106C:"Bad password, try again\n" rcx:ZwQueryInformationThread+14
--	--

So now I can see Assembly of the program in disassembled section. We need to read this section to understand what's the program do under the hood. So, we can see our password named "CrackMeOne" and in the disassembled section we can see something like "**lea rdx, qword ptr ss:[rbp-8]**". Under this line we can see "**jne acrack.7FF706CC15BD**". So, **JNE** means in assembly is **Jump if Not Equal**. We can see same for our Password.

So that's meaning is If our password doesn't match, Jump to "acrack.7FF706CC15BD". So, let's change the condition. What if we can change the **JNE** to **JE?** **JE meaning is Jump If Equal.** So will go to that line and press space button to assemble that line.

<pre> lea rdx,qword ptr ds:[7FF706CD1042] mov rcx,rcx call <JMP.&strcmp> test eax,eax je acrack.7FF706CC15BD lea rax,qword ptr ds:[7FF706CD104D] mov rcx,rcx call acrack.7FF706CC14A4 lea rax,qword ptr ds:[7FF706CD1054] mov rcx,rcx call acrack.7FF706CC14A4 mov rax,qword ptr ss:[rbp-10] mov rdx,rax lea rax,qword ptr ds:[7FF706CD103B] mov rcx,rcx call acrack.7FF706CC1450 mov rax,qword ptr ss:[rbp-10] lea rdx,qword ptr ds:[7FF706CD105F] mov rcx,rcx call <JMP.&strcmp> test eax,eax je acrack.7FF706CC15AC lea rax,qword ptr ds:[7FF706CD104D] mov rcx,rcx call acrack.7FF706CC14A4 jmp acrack.7FF706CC15CC lea rax,qword ptr ds:[7FF706CD106C] mov rcx,rcx call acrack.7FF706CC14A4 jmp acrack.7FF706CC15CC lea rax,qword ptr ds:[7FF706CD1085] mov rcx,rcx call acrack.7FF706CC14A4 mov eax,0 add rsp,30 pop rbp ret </pre>	<pre> 00007FF706CD1042:"CrackMeOne" rcx:ZwQueryInformationThread+14 00007FF706CD104D:"True!\n" rcx:ZwQueryInformationThread+14 00007FF706CD1054:"Password: " rcx:ZwQueryInformationThread+14 00007FF706CD103B:"%s[\n]" rcx:ZwQueryInformationThread+14 00007FF706CD105F:"t.me/Umn0223" rcx:ZwQueryInformationThread+14 00007FF706CD104D:"True!\n" rcx:ZwQueryInformationThread+14 00007FF706CD106C:"Bad password, try again\n" rcx:ZwQueryInformationThread+14 00007FF706CD1085:"bad nickname! try again\n" rcx:ZwQueryInformationThread+14 </pre>
---	---



So, I just re assembled the **JNE** to **JE** both of lines. And now need to patch this application. So, I will press **CTRL+P** to patch the application. Then patch the application and save new application as **cracked.exe**. Let's execute our new program.

07FF706CC153D 48:8D15 FEFA0000 lea rdx,qword ptr ds:[7FF706CD1042] 00007FF706CD1042:"CrackMeOne"
07FF706CC1544 48:89C1 mov rcx,rax rcx:ZwQueryInformationThread+14

C:\Windows\System32\cmd.exe

C:\Users\Piyusha Akash\Desktop\RevEng>dir *exe
Volume in drive C has no label.
Volume Serial Number is 26D1-2522
Directory of C:\Users\Piyusha Akash\Desktop\RevEng
12/15/2025 07:25 PM 383,745 acrack.exe
02/03/2026 06:27 PM 383,745 cracked.exe
2 File(s) 767,490 bytes
0 Dir(s) 38,245,040,128 bytes free

C:\Users\Piyusha Akash\Desktop\RevEng>cracked.exe
Mini crackme by @microsoftstorage wriited on C!
Nickname: .
True!
Password: .
True!

C:\Users\Piyusha Akash\Desktop\RevEng>

So, as you can see, we will successfully patch the application. Now it will work for any credentials. In that case I put “.” As my Nickname and Password. And it's working now. So, what we did? What happened?

We will change the logic in assembly. In assembly this happened. If we entered wrong credentials, Logic jumps to bad credentials message. If it is correct, Logic goes normally. That's why we change **JNE** to **JE**.

Thank you for reading this writeup. Don't forget to follow me. Best regards, **Piyusha Akash. (0x3xp)**

LinkedIn: <https://linkedin.com/in/piyushaakash>

GitHub: <https://github.com/0x3xp>

YouTube: <https://youtube.com/@infoseck>

TikTok: <https://tiktok.com/@infoseclk>

Linktree: <https://linktr.ee/piyushaakash>